Issues for a “Do Not Email” List

Matt Bishop

Department of Computer Science[1]
University of California, Davis
1 Shields Ave.
Davis, CA 95616-8562

bishop@cs.ucdavis.edu
phone: (530) 752-8060

1. Introduction

The ubiquity of the problem of spam is evident to anyone who uses a computer. Unsolicited electronic email carries offers for items such as drugs, software, or sexual aids, and nasty attachments such as Trojan horses, computer viruses, and computer worms. Numerous approaches to handling spam have failed. There is little reason that spam will decrease.

The “do not call” telephone list has been remarkably successful in reducing the number of calls that people receive from telemarketers. The FTC is considering having a “do not email” (DNE) list, similar to the “do not call” list. A Request for Information was issued.

This report begins with a discussion of what spam is. It then summarizes the goals of a DNE list. It then discusses three models of administering the DNE list, and several issues central to such a list. We conclude with a discussion of the key points underlying difficulties in selecting appropriate technical mechanisms.

2. Definitions

The purpose of a DNE list is to reduce the amount of spam that people receive. However, the exact definition of spam is unclear. The title of the CAN-SPAM Act defines it as “non-solicited pornography and marketing”. Traditionally, there has always been an element of mass e-mailing in the definition of spam. Thus, a single unsolicited letter of marketing matter or pornography, sent to a single recipient, may be unwelcome, but it is not spam in the traditional sense. Yet if the same letter were sent to many thousands of people, it would be considered spam.

In this report, we distinguish between two types of unsolicited email. UCE, or unsolicited commercial email, is email that advertises a product or service (pornographic or otherwise) and is sent to a large number of people. This is the traditional definition of spam.

The second type of unsolicited email is more pernicious and dangerous. This email is an attempt to trick the recipient into compromising herself or her system. For example, some contain an attachment designed to send copies of itself to everyone in the recipient’s address book, thereby flooding the networks with email (a *computer worm*); some contain an attachment designed to delete files on the recipient’s computer, or to create back doors through which the senders can break in (a *Trojan horse*); some contain an attachment that inserts itself into files on

1. Affiliation included for identification purposes only. The opinions expressed in this paper are those of the author alone, and not necessarily the opinions of any other person, entity, agency, or organization.

the user's system and damages them (a *computer virus*); and some contain other forms of this malicious content that is triggered by the recipient activating the attachment (or, under some mail readers, by simply reading or selecting the letter[2]). Others, which do not have attachments, direct the recipient to click on a hyperlink and enter her name and password to preserve a bank account, for example—except the link goes to the thief's site, thereby tricking the recipient into giving the thief a user name and password for a legitimate bank account (a *phishing attack*). These are not "commercial" email messages in the traditional sense of "commercial", so the term UCE does not apply. But such letters are typically sent to many thousands of addresses, and constitute a serious problem akin to UCE. We refer to these as BUE, for bulk unsolicited email.

In this report, the term "spam" refers to both UCE and BUE.

3. Goals of a DNE List

The purpose of the DNE list is to reduce the amount of spam. To this end, there are four specific goals that pertain to any implementation of the DNE list.

1. The e-mail addresses on the DNE list must be used only to determine who *not* to send spam to. If the list is made available for other purposes, or is used for other purposes, it will provide a source of confirmed email addresses for any spammer to use. In practical terms, this means that the list must be kept confidential from the public. It also means that, if the list is made available to marketers, every person in that marketing group that sees the list has to be trusted not to divulge addresses on the list, not to make the list available to others, and not to use the list to send spam personally.
2. The e-mail address on the marketers' lists must be used only to determine who to send spam to. This flip side of the previous goal is to ensure that a marketer does not make its mailing list available to others when using the DNE list. That way, a marketer does not risk its competitors acquiring the marketing list and gaining a competitive advantage.
3. All mass e-mailers must use it. Otherwise, the amount of spam *may* decrease because of the commercial vendors who use the list, but the senders of spam who do not use the list will not decrease their output. Indeed, overall the amount of spam could still increase.
4. The use of the DNE list must not change the infrastructure of the Internet. First, the Internet is a global network, and only those portions within the United States are subject to U. S. law. The U. S. cannot enforce its required changes upon parts of the Internet not under its jurisdiction, and enforcing changes only upon the U. S. infrastructure would effectively destroy the collaborative environment that the Internet relies upon. Further, changing protocols will cause existing services to break, or degrade. This is why the Internet Engineering Task Force requires much testing, analysis, and discussion to advance proposed protocols and protocol changes to standards. Finally, assuming these two problems could be overcome, the problem of providing software incorporating the changes and that would be universally accepted seems insurmountable.

These goals will not solve the problem of an *authorized* user using the DNE list for *illegitimate* purposes. Suppose, for example, The Electronic Marketing Co. is authorized to use the DNE list, and obtains a set of addresses on that list. It deletes those addresses from its email lists,

2. For a more detailed description of the types of programs that can cause problems, see for example Chapter 22, "Malicious Logic," of Bishop [1].

thereby complying with the terms under which it may use the DNE list. But Robin, the employee with the job of culling the DNE addresses from the company's list of addresses, goes home and contacts Spam Pour Vous, a foreign company that sends spam to email addresses in the United States. Robin sells them the addresses that the Electronic Marketing Co. has acquired. Robin is an authorized user, authorized to receive the DNE list to strike names from a company mailing list. But Robin is using the addresses also for an illegitimate purpose: to provide a spammer, outside the jurisdiction of the United States, with a set of valid, confirmed, email addresses in the United States. Likewise, illegitimate marketers, who register to obtain authorized access, can use the DNE list to obtain valid addresses. There are no technical means to prevent this type of subversion. Some means to detect it will be discussed below.

4. Models of Distribution

Assuming the DNE list is to be kept confidential (in the sense of goal 1), there are three models describing the way in which vendors' email lists can be processed against the DNE list. Some preliminaries will make the discussion of the models simpler.

4.1. Preliminary: Storage of the List

Common to all models is the representation of the DNE list. The DNE list is composed of elements which must include the email address, and may include other elements such as associated name, the date the entry expires, and other information. Presumably, the email addresses are the only components considered for distribution. A naïve approach is to keep the email addresses in cleartext form. The problem is that anyone who acquires access to the list, licitly or illicitly, will obtain valid email addresses, which they can proceed to send spam to. Some mechanism to detect vendors abusing the list is needed. One way is to plant regulatory email addresses, called *canary addresses*, in the DNE list, on the theory that the only email they will receive will come from spammers who acquired the DNE list. Section 8.5 below discusses this method, and the problems with it, in detail.

A more sophisticated approach is to encipher the addresses, and give cryptographic keys to those authorized to read the list. But this suffers from a similar problem, because when the list is used, it must be deciphered. This makes the cleartext addresses visible to the users, and to any unauthorized persons who have access to the system on which the list is deciphered. The issue of use of the list is still a problem.

An alternate approach follows that of password storage. Cleartext addresses are *never* stored with the list. Instead, each address is transformed (*hashed*) using a cryptographically strong one-way function (called a *hash function* in this paper). The DNE list consists of the result of the hash function applied to each email address (each result is called a *hash*). Such a function is not invertible and the result cannot be decoded to find the original email address.[3] So the attacker who acquires the list cannot decode it, to determine the addresses on the list.

However, the vendor is able to determine whether a given email address is on the list. The vendor hashes the given email address, and checks to see if the resulting hash is on the list. If it is,

3. More precisely, a widely accepted but unproven hypothesis in the theory of computation (specifically, that $P \neq NP$) would need to be false for such functions not to exist. In practise, hash functions such as SHA-1 [4], MD5 [6], and others are considered non-invertible. They also meet the other criteria for a cryptographically strong hash function (see for example [1], pp. 237–238).

the given email address is on the DNE list and nothing should be sent to it. If it is not, the given email address is not on the DNE list. Note that this also tells an authorized spammer, who has obtained authorized access to the DNE list, that the address is on the DNE list and is therefore likely to be valid. In other words, lists containing email addresses, enciphered email addresses, and hashes all suffer from the same problem: any authorized user of the DNE list can check for valid addresses.

In many contexts, the use of hash functions is effective in thwarting attacks. But in this context, its effectiveness is questionable because a spammer can do what a vendor can do. If the spammer has the list of hashes, and wants to determine whether an address is on it, all the spammer needs to do is compute the hash of the address, and see if the hash is in the list. This is called an *off-line dictionary attack*. For example, using SHA-1, an attacker can test approximately 450,000 addresses per second on a very simple computer system.[4]

To speed the attack, an attacker could generate a list of addresses, hash each once, and compare the resulting list to the stolen list. So, for example, if the attacker wishes to determine which of 1,000 addresses are on a list of 3,000 hashes, she needs to compute 1,000 hashes and do 3,000,000 comparisons in the worst case. On the system cited above, this would take slightly more than 2 milliseconds.

Salting increases the computational difficulty. A random string of characters (the “salt”), different for *each* address, is generated and combined with the address to produce the hash. The salt is stored with the hash. Then, an attacker will need to hash each address on the list once *per entry on the hashed list* to run the dictionary attack. Returning to our example, as each of the 1,000 hashes has a different salt, the list of addresses to be guessed must be hashed 1,000 times (once per salt), for a total of 100,000 hashes and 300,000,000 comparisons in the worst case. This takes effectively 1,000 times as long as the case where salting is not used. On our example system, this would take about 2 seconds.

Salting is used in other contexts, especially in the protection of passwords [3]. As the salt is needed to determine whether a guessed password is correct, the salt is usually stored with the hash itself. Separating the salt from the hash gains little in terms of security, because both the salt and the hash must be available to the system to validate a supplied password. The overhead of having to keep track of which salt is associated with which hash, plus of having to retrieve both to validate a password, outweighs the risk of someone who illicitly obtains the hashes also obtaining the salts. A similar argument applies to hashes and salts when email addresses rather than passwords are being protected.

But the analogy with passwords breaks down because each password hash is associated with a user. When a user enters a password, the salt and password are combined to produce a hash, and exactly one entry is checked (the entry associated with that user). There are no names associated with the email addresses stored as hashes; indeed, doing so would defeat the purpose of storing the hashes of the addresses! But this means the repository must check each email address to be validated against *every* hash in the DNE list. So salting slows the scrubbing of submitted lists, too.

In the discussion of the models below, when we write of a vendor determining which addresses are on the DNE list, we mean that either the vendor sees the addresses in cleartext (if

4. This figure was obtained using an off-the-shelf computer system running the freely available operating system FreeBSD 5.2, with a 200 GHz Intel® Pentium® 4 chip.

the addresses in the DNE list are not hashed), or uses the dictionary method to determine that some set of addresses has hashes that are on the distributed version of the DNE list (if the addresses in the DNE list are hashed). When we write of addresses in the DNE list being available, we mean either the cleartext addresses (if the addresses in the DNE list are not hashed) or the hashes of the addresses (if the addresses are hashed).

4.2. Preliminary: Contents of the List

Assume the DNE list contains hashes of the entries. If the entries are email addresses only, the vendor's hashing of email addresses poses no problem. But if the entries are both email addresses and domain names, the issue of how to handle vendors' email lists when those lists are to be hashed poses a problem. The problem is that the result of hashing a substring of a string bears no relation to the result of hashing the string. So, if domain names are allowed in the list, then the checking requires a two-step process. First, the email address is hashed and compared to the DNE list. Then each email domain (the part after the "@") must be hashed and compared to the DNE list. This task can be made less onerous by marking each entry in the DNE list as an "address" or a "domain name". In what follows, we assume this issue is handled appropriately for the type of list and entries.

4.3. Model #1

In this model, the DNE list is distributed to marketers. The marketers then determine which addresses on their mailing lists are also on the DNE list, and remove them. Updates to the DNE list are pushed out to the marketers, or the marketers must periodically request updates to the DNE list.

The primary advantages of this model are simplicity and confidentiality to the vendors. One architecture of systems following this model is to place the DNE list on a repository (which may be centralized or distributed) and give vendors access; the other is to give each vendor who agrees to use the DNE list a copy of that list. Updates may also either be made available on repositories, or sent directly to the vendors. This satisfies goal 2, because each vendor keeps its own list and need never show it to any other entity. Further, the architecture requires only a set of systems to administer the list (add people, update entries, and provide access for the vendors).

The disadvantage to this model is the lack of confidentiality of the DNE list. Even assuming communications between the vendor and the repository are secured (to preserve the confidentiality, integrity, and availability of the DNE list), the addresses in both the DNE list and the vendor's list are available to the people working for the vendor. Worse, if the vendor wanted *not* to abide by the rules, she would have a list of valid email addresses to use (if the list is not hashed) or a list of hashes to test (if the addresses on the DNE list are hashed). This disclosure of the elements of the DNE list violates goal 1.

4.4. Model #2

In this model, the DNE list is held at a central repository, or is distributed to trusted repositories. The marketers then send their list of addresses to such a repository. That repository returns the list with addresses on the DNE list removed. As an alternative, the repository may return a list of addresses on both the DNE list and the marketer's list, so the marketers may remove those addresses. The two approaches are equivalent because the marketer can compare the sent copy of the list with the one returned, and determine which addresses were deleted (and therefore on the DNE list).

The advantage of this model is concealment of the DNE list. The vendors never acquire a copy of it. Hence they cannot determine which addresses are on the list without submitting a list of addresses to a (presumably trusted) repository. Further, the repository can guard against dictionary attacks by monitoring the frequency of list submission and the nature of the addresses on the lists. If the marketer submits lists every 10 minutes with no overlap, the repository may consider that evidence of a dictionary attack. If the marketer submits lists with no overlap and very large numbers of names, that might indicate an attempt to try to find names on the list (although it may also be evidence of an active e-mail marketer, and not indicate anything untoward). More study is needed to discover the anomalies that would indicate an attempt to compromise the list in this situation and to understand how spammers might try to evade them. This model satisfies goal 1 to the extent of protecting the full contents of the list, because the addresses on the DNE list are never disclosed to vendors except when the vendor has one of those addresses on the marketing list. But it does not keep *all* addresses on the DNE list confidential, as the vendor can determine which addresses are on both the DNE list and her list.

This method does not satisfy goal 2 because the vendors must submit their lists to the repository. If the repository is trusted, then there is no harm as the lists will not be made available to anyone other than the system checking the addresses against those on the DNE list. But whether the marketers will trust the repository is another issue, one beyond the scope of this technical paper. One way to ameliorate this problem, but at greater cost, is to provide the marketers with the set of salts used to produce the hashes in the DNE list. The marketers would then submit their lists of addresses as sets of hashes. The repository would then compare lists, and send back the set of hashes in (or not in) the DNE list. This protects the confidentiality of both lists as much as is possible (in the sense that both the vendor and the repository will know addresses in both lists, but neither will know any addresses in the other's list but not in theirs). The problem with this method is the cost in computation; if the DNE list has 1,000,000 entries, the vendors must compute 1,000,000 copies of their list (one copy per salt, assuming each address in the DNE list has a unique salt) and send all lists to the repository. This appears infeasible in practise.

4.5. Model #3

In this model, the DNE list is held at a central repository, or is distributed to trusted repositories. Unlike Model #2, however, the repositories are also mail forwarders. The vendor sends out email containing the spam, but sends it to one of the repositories. That repository simply eliminates all recipients on the DNE list, and then forwards the email to all the others.

Like Model #2, the addresses on the DNE list are concealed from the marketers, and all the advantages of Model #2 apply here. So, this model satisfies goal 1. Further, the marketer never sees the list of email addresses to which spam is not forwarded, so in this sense—that the vendor never learns which of the addresses on the vendor's list is on the DNE list—the confidentiality of the DNE list is more strongly protected than in Model #2.

The disadvantage of Model #2 still applies. The vendor must trust the repository not to disclose the list of recipients to other vendors, so this mechanism does not satisfy goal 2. Further, the vendor cannot submit hashes of her list, because the repository cannot forward the email to hashes. Indeed, the very nature of this model requires that vendors disclose their mailing lists to the repository. Further, the repositories must bear the burden of forwarding the mail, and if the vendors' lists are large, this will take considerable resources. Hence the need for a set of repositories, each with a copy of the DNE list, and each with the computing power and network connectivity to forward large quantities of email, is a critical issue.

4.6. Analysis

If *all* entries of the DNE list are to be kept confidential, then only Model #3 is acceptable. The problem is that if the vendor is told not to send email to a set of addresses, the vendor may assume, with a high degree of accuracy, that those addresses are on the DNE list and hence are valid. Presumably, the vendor will keep such addresses in a special list, to check newly obtained addresses against them to reduce use of the repository (this is especially true if the repository charges a fee for each address validation). The addresses in this list could then be compromised or misused, by the vendor or by others.

Similarly, if *all* addresses of the vendors' lists are to be kept confidential, then only Model #1 is acceptable. Both other models require the addresses to be disclosed. Model #3 requires that the raw addresses be sent to the repository. Under Model #2, the vendor may submit hashes, but if any match hashes on the DNE list, the repository will be able to determine some addresses on the vendor's list. Further, the procedure of hashing the lists once per salt, and checking them, will be very expensive.

The appropriateness of the model to be used therefore depends on the policy selected with respect to confidentiality of the lists. If protecting the contents of the DNE list is paramount, Model #3 is clearly the best. If protecting the contents of the vendors' lists is paramount, Model #1 is clearly the best. But one suspects factors such as cost and ease of implementation and maintenance will affect the selection of the actual policy to be used, and in that case, Model #2 may turn out to be the best. In short, the policy requirements must guide the resolution of the technical issues.

Each of these models has its flaws. Model #1 discloses the DNE list to vendors. Model #2 and Model #3 disclose the marketer's lists to the repository. Model #3 also will require great network connectivity to handle the sending of the letters. The acceptability of each problem in the models is a decision for the policy makers.

5. Requiring Use

Unfortunately, there is no technical way to achieve goal 3. The problem is that *anyone* can acquire email addresses, and anyone can email to any Internet email address. Given the architecture of the Internet, it is simply not possible to force all email to flow through central servers to vet addresses. The Internet was designed to be robust in the face of widespread failures, and funneling all SMTP (email) traffic through some relatively small set of hosts would eliminate this. It would also create congestion in the networks, clogging key portions of the Internet and slowing the delivery of email to an unacceptable extent. As recent estimates are that *31 billion* pieces of email flow through the Internet daily[5], avoiding this congestion would require thousands, if not tens of thousand or more, servers to check and forward the mail—and the management of these servers would be a major security problem, Further, the architectural and infrastructure changes required to implement such a funneling would not be widely accepted and could not be enforced.

5. This estimate is taken from http://www.spamfilterreview.com/spam-statistics.html on May 11, 2004.

Even assuming the above could be done, though, how does one determine whether a given email message is spam? Section 2 defines spam as UCE or BUE. But if Anne sends a letter to Bill, how can the central systems (in the sense discussed in the above paragraph) determine if Anne's letter is one of a larger mailing that qualifies as UCE, or contains malicious content that would qualify it as BUE? The need to do the former implies the ability to monitor the contents of all emails, something that is clearly technically infeasible and repugnant to most societies. The latter implies the ability to analyze the bodies of all email, again requiring them to be read. From the technical point of view, the email would either need to be held while the analysis was conducted (which would create massive delays, not to mention extreme congestion about the central funneling systems) or email would be deleted erroneously, because of the emphasis on preventing possible spam. Again, from the technical point of view of keeping the Internet functioning, there is no known, workable solution to the problem of identifying any individual letter as spam.

What makes the above even more untenable is that what is "spam" to one individual may not be "spam" to another. The author periodically received BUE. Normally, he considers it spam, and discards it. But recently, some of his students and co-workers have begun a project requiring examples of a specific type of malicious content. The author no longer considers them spam in the sense of BUE. They are "research data" that he promptly forwards to his students and colleagues working on that project. As another example, if one receives a letter from the RIAA claiming that the individual is suspected of downloading copyrighted data, and the individual has not done so, does that qualify as "spam"? The RIAA would think not, but the individual might well think so. How would central nodes handle these cases? Absent some universally accepted policy (which seems infeasible because peoples' opinions differ), there is no technical solution here.

6. Integration with the Internet

As indicated by goal 4, and discussed further in section 5, implementation of the DNE list must not rely upon changes to the Internet. It may encourage changes, or use new protocols, but must recognize that these changes and new protocols will *not* be adopted quickly. Further, because protocol design, and changes to the infrastructure, almost always have serious, unexpected side effects, any such proposed changes or protocols must be scrutinized carefully, tested, the results of the tests analyzed, changes made as needed, testing repeated under a variety of conditions, and then the changes and protocols must be slowly deployed. The process for this involves the Internet Engineering Task Force (IETF), a body that analyzes proposed protocols from their proposal to their deployment. The IETF is a grass-roots organization, and the concerns of its members are varied, but all want to keep the Internet functioning. So, any changes must be carefully thought out and developed with the engineering components a critical consideration.

A second issue is how to integrate proposed changes with existing software. If the DNE list requires specific parties, such as vendors currently sending spam, to use software to interact with repositories, the software must take into account the multiplicity of systems used today. Developing software for Microsoft systems, for example, would prevent any vendor who used Linux-based systems, or Sun Solaris systems, from using the list (to cite two examples). The interactions with the DNE list must be devised to allow *any* vendor who wants to use that list to be able to do so. If existing software is to be modified, the software vendors must be persuaded to do so, and do so in a way that does not break their existing software. This technical issue will undoubtedly affect the selection of the policy for the DNE list.

Third, software and proposed changes must maintain backwards compatibility with existing software and protocols. The reason is that many sites will simply not adopt the changes. The commercial world has proven this, with (for example) Microsoft's maintaining backwards compatibility on its Windows platforms. It applies equally well in other areas. For example, SMTP, the Internet's primary electronic mail transport protocol, was developed by 1982 [5]; its latest incarnation (from 2001) does not "add new or change existing functionality of … the original SMTP (Simple Mail Transfer Protocol) specification of RFC 821" ([2], p. 1). This ensures that systems that are not upgraded will continue to interact with systems on the Internet.

Ensuring backwards compatibility is a difficult task because the correct functioning of systems and software depends upon assumptions that developers make about the environment in which the systems run. It requires considerable care and effort, and co-operation from those who do not wish to upgrade to the new mechanisms as well as those who do. Sometimes, people choose not to co-operate, or may not realize that they need to consider the upgrade.

For these reasons, requiring changes to the Internet to implement a DNE list is a poor use of resources from the technical point of view, unless one accepts that portions of the Internet users (vendors) will not use the DNE list, and unless one considers isolating portions of the Internet for some time (possibly permanently) is acceptable.

7. Other Issues

Several other issues affect the technical aspects of a DNE list but did not fit in elsewhere. They are summarized here.

7.1. Attackers and Systems With the DNE List

It is vital that the proposers, designers and implementers of the DNE list understand that *any computer system can be compromised; security is a matter of costs and benefits, and is not an absolute*. The goal is to make the cost of breaking into a system greater than the benefits that the successful attacker will reap. This makes maintaining a cleartext list of addresses on the DNE list questionable, since even if attackers get the list, they will need to guess addresses rather than discover the set of cleartext addresses that are known to work (as they registered successfully for the DNE list). Further, if cleartext addresses are used, they might be accidentally exposed. Hence the addresses should reside on systems only in hashed form.

Sadly, the situation is worse than might appear. If consumers register using a web-based system (as seems likely), then in the period of time between the address being entered and it being hashed, that address is potentially visible to any attacker who gains access to the system. The developers of the software, systems, and procedures administering the DNE list must consider minimizing the threat posed by someone who breaks into the system as well as the threat of someone breaking into the system. The standard technique for this is a layered defense (called "defense in depth" in government, and an application of the principle of separation of privilege [7]).

The web-based interface that consumers use to register email addresses, and the interfaces that vendors use to submit their lists of addresses under Model #2 and Model #3 are also gateways to the system that must be protected. Presumably, these interfaces will be implemented using network-based servers. Such services have a long history of being attacked, and being attacked successfully. The implementers should assume that the interfaces and their supporting services will be attacked. They should have a layered defense to handle cases where the attacks succeed.

7.2. Dictionary Attacks on a Hashed DNE List

Although this has been alluded to in section 4, there are some aspects of this worth emphasizing. Recall that a dictionary attack consists of an attacker guessing an address, computing its hash, and then looking for that hash in the DNE list. (If the DNE list is composed of cleartext email addresses and domain names, this section is irrelevant.) If an attacker can obtain the list of hashes, then the attacker can launch a dictionary attack without fear of detection using her own computers (this is called an *off-line dictionary attack*). However, if the attacker cannot obtain this list, then the only type of dictionary attack available is to supply email addresses to a repository, and see if any of them are marked as being on the DNE list. As mentioned in section 4.4, more study is needed to determine whether this type of attack can be detected.

7.3. Denial of Service Attacks

A reasonable question to ask is: what happens when the DNE list is not available? If Model #1 is used, the marketers will not be able to receive updates; if either of the other models are used, the repositories will be inaccessible and the vendors cannot validate their lists. In all three cases, how do consumers add their names to the DNE list should the administrative system(s) be unavailable? A distributed denial of service (DDoS) attack on the repositories could easily create this situation.

A distributed denial of service attack may occur in many ways. First, a small set of users (or automated attack tools) could add random names to the DNE list. If the repositories cannot handle the volume of traffic, then legitimate consumers will be unable to access the service that adds names to the DNE list. The volume of traffic could be made as great as desired, especially if the attackers deposit tools to flood the repositories on systems that they break into. Also, depending on how addresses are validated, the amount of computation and network messages used to do the validation might overwhelm the system. Secondly, automated attack tools could simply flood the network connections into the repositories, preventing any legitimate network traffic from arriving. This would isolate the repositories. The attacks on Amazon and E-Bay a few years ago attest to the effectiveness of the attacks.

Unfortunately, because of the structure of the Internet, there are currently no effective ways to prevent such a denial of service attack. The impact of this fact upon the DNE list models is that the models assume the repositories are available to the vendors at all times. Model #1 requires that the DNE list be sent over the network to the vendors, and that the repositories be available so the vendors can poll them for updates. Model #2 requires that the vendors and the repositories be able to exchange messages. Model #3 requires tremendous bandwidth to forward the letter to the addresses not on the vendor’s list. If the network, or the repository, is unavailable, the vendors cannot use DNE list. So a denial of service attack is a serious threat to the DNE list.

Researchers are investigating ways to ameliorate their effects, and to develop mechanisms and protocols to defeat such attacks. But these will not come to fruition in the short term, so the developers of the DNE list must be prepared to deal with such attacks, and take measures to mitigate them.

7.4. Centralized vs.Distributed

Consider the models in which the vendor wants to check a list of addresses for addresses on the DNE list (Model #1 and Model #2). The question of whether the DNE list should be available from a central server, a set of distributed servers, or a set of satellite servers affects the resiliency of the DNE list to denial of service attacks. A centralized server would have the DNE list on one host, which would be a single point of failure. Congestion in network traffic would also be a problem, as all vendors and customers would access the same server. Backup servers would ameliorate this problem, but the backup servers should be geographically (and logically) separate from the primary server. A second architecture is to maintain the list at a central site, but push copies of it to satellite servers that vendors could use. This distributes the load among various sites across the Internet, avoiding the congestion problem of a central server, and providing redundancy in case of a failure of the central server (or any of the satellite servers). Security is still a major concern; indeed, the distributed mechanisms increase the security risks (the degree depending on the specific mechanisms used). The reason is that the co-ordination, synchronization, and maintenance of the distributed systems is more difficult to perform than those functions on a centralized server.

A distributed architecture would involve using a distributed database to contain the list. This introduces problems of synchronization, updating, and consistency. Various techniques solve these problems.

Technically, it is unclear which design would work best. It depends upon the specific environment and procedures for maintaining the systems that implement the DNE list.

7.5. Auditing Use of the Lists

If vendors are given access to the lists in such a way that they can determine any (or all) of the addresses on the lists, some mechanism must be in place to detect abuse of the list. One way to do this is to include addresses that the regulators will monitor. These addresses are called *canary addresses* or simply *canaries*. Each vendor's copy of the list is seeded with a different set of canaries. If email arrives at any canary, the regulators can tell which vendor is sending email to addresses on the DNE list.

The problem with this idea is that spammers may locate the canary address without referring to the DNE list, in which case a vendor will be erroneously accused of leaking the address. Spammers locate addresses in a wide variety of ways, including asking mail transfer agents for addresses to which they will deliver and generating random or pseudorandom guesses for account names followed by known domain names. Further, a vendor's system may be compromised and the addresses obtained sold. Finally, the system on which the DNE list resides, or corrupt regulators with access to the information on the list, may compromise the confidentiality of the list. So a canary address receiving email *may* indicate a problem with a vendor misusing the DNE list addresses, but is by no means conclusive proof.

8. Conclusion

The effectiveness of a DNE list raises several problems. The greatest problem is defining "spam" meaningfully. The second is defining the class of emailers who must use it. The third is catching vendors who must use it but do not. The fourth is measuring the effectiveness of the DNE list.

For purposes of this paper, spam was defined to be unsolicited commercial email (UCE) or bulk unsolicited email (BUE). However, as section 2 points out, the definitions are nebulous. Political parties have sent email to many thousands of recipients simultaneously. If one views the letters as "selling" a candidate or a political platform, are the letters spam? If the Catholic Church, or Jerry Falwell's ministry, or People for the American Way, sends a mass mailing urging people to pray for peace, is that spam? If I send a letter selling Matt's Miracle Medicine to one friend, that probably is not spam under any reasonable definition (because personal email to a friend is not spam). But if I send it to 10 people, is it spam? To 100 people? To 1,000 people? At what point does it become spam? The definition is imprecise, and in fact different for each individual. The relevant law defines spam to some degree, but the regulators may have to interpret the law in order to apply it to particular emails.

The second problem, implicit in the first, is to define the class of senders (called "vendors" or "marketers" throughout this paper) required to use the DNE list. Simply saying that people who send bulk email must use the list raises the question of what "bulk" email is. Exempting classes of people, such as political parties or churches, raises the issue of what a "political party" or a "church" is. Defining senders on the basis of recipients, such as "those who send emails designed for children ages 5–10", is even more difficult because identities are not associated with email addresses; for example, the author's son sometimes uses his father's personal account to send and receive emails, and the author's grandson uses his mother's account (as he is 2 years old, his mother typically writes the letters and reads them to her son). So, who is the person associated with those accounts? This is why associating an identity with an email account is tricky; the account may be shared, or the class of consumers using the email address may vary over very short intervals of time, or email addresses may be monitored by supervisors or parents.

Complicating this problem is the international, global nature of the Internet. The author is not a lawyer, but it is his understanding of basic legal principles that United States law does not apply to people in Bulgaria, Hungary, Russia, or other countries in general. Certainly United States laws are not enforces in those countries. So, if a vendor from those countries sends spam to email addresses whose readers are in the United States, should the vendor be required to use the DNE list? If so, how does one enforce that requirement? There is no technical mechanism that can do so. The relevance of this is clear when one realizes that a multinational corporation obtaining a set of addresses on the DNE list may have its foreign subsidiary send the spam to those addresses with seeming impunity. How will the DNE list apply to spam originating from abroad?

The root of this problem is itself a third, different problem. There is no meaningful authentication in the Internet.[6] One cannot prove the origin of an ordinary piece of email, and all spam is ordinary email. So, how can one trace offending messages back to their source? There are two parts to this problem. The first is authenticating the originator, so that an independent third party can verify who sent it. The second is verifying the *location* of the sender, to ensure they are subject to the relevant laws and regulations. Given the mobility of people in the world today, the ability to use a myriad of email addresses, and the ability to hook a computer up to the Internet practically anywhere in the world, this is a non-trivial problem. By the time email is traced back to its origin[7], the sender may have disappeared.

One obvious solution seems to be to require all spam to be watermarked or digitally signed by its originators. Enforcing this would require any such email without the watermark or digital signature to be discarded automatically. But how does an ISP distinguish between spam and non-

6. In technical terms, simple (and widely known) techniques allow anyone to forge the originating address of an email message, and with a bit more skill and luck forge IP addresses. This forging, or masquerading, as another entity is called *spoofing*.
7. This assumes that such a trace is accurate and successful. Neither assumption may be true.

spam? As discussed above, this is infeasible. Taken to its extreme, one might think to require *all* mail to be digitally signed or watermarked, and any mail not so signed or marked is to be discarded. This would mean that every user on the Internet must have a cryptographic key for the signature, and a ubiquitous key distribution and validation infrastructure must be in place, or some similar mechanism for watermarking and validating watermarks. This would require that the Internet be re-engineered. Further, we have been unable to develop a widely used, generally accepted public key infrastructure; the mechanisms needed to do what is discussed above are considerably more complex, as public key cryptography is *not* considered acceptable in many quarters, especially by governments and government agencies that desire to be able to read enciphered electronic mail.

This leads to a fourth problem, ease of use. This report did not discuss issues such as how to register consumers and vendors, revocation and expiration of registration, and so forth. Experts in other areas (such as human factors) can provide information on the best way to handle these problems. From the point of view of security, the principle of psychological acceptability [7] requires that these matters be considered, and that the mechanisms adopted be as simple and straightforward to use as possible.

The final problem is how to measure the effectiveness of the DNE list. One issue is the precise *purpose* of the DNE list. Is it designed to lessen the flow of spam around the Internet, or to limit the amount of spam that individual email addresses receive? or is its primary purpose some mixture of the goals in section 3? If so, in what proportion? What are the specific criteria for judging the success or failure of the DNE list? Any serious proposal to implement a DNE list must describe a method for gauging the success of that list, so the relevant authorities can determine what needs to be changed should the DNE list not achieve its purpose.

Other issues, such as models for architecture, integration into the Internet infrastructure, and impact upon the Internet infrastructure, have been discussed. All raise serious technical issues. Solutions must be guided by policy considerations.

A "do not email" list raises many interesting technical, social, and political questions. This paper has presented the main technical issues, and touched on others. Their resolution is left to the developers of such a list, who must take into account the technical and non-technical environment in which such a list will function.

9. References

[1] M. Bishop, *Computer Security: Art and Science*, Addison-Wesley Publishing Company, Boston, MA (2003).

[2] J. Klensin, *Simple Mail Transfer Protocol*, RFC 2821 (Apr. 2001).

[3] R. Morris and K. Thompson, "Password Security: A Case History," *Communications of the ACM* **22**(11), pp. 594–597 (Nov. 1979).

[4] National Institute of Standards and Technology, *Secure Hash Standard*, FIPS PUB 180-1 (Apr. 1995).

[5] J. Postel, *Simple Mail Transfer Protocol*, RFC 821 (Aug. 1982).

[6] R. Rivest, *The MD5 Message Digest Algorithm*, RFC 1321 (Apr. 1992).

[7] J. Saltzer and M. Schroeder, "The Protection of Information in Computer Systems," *Proceedings of the IEEE* **63**(9), pp. 1278–1308 (Sep. 1975).